Fast, Accurate Simulation for SDN Prototyping

Mukta Gupta
University of Wisconsin
mgupta@cs.wisc.edu

Joel Sommers
Colgate University
jsomers@colgate.edu

Paul Barford
University of Wisconsin
pb@cs.wisc.edu
Motivation

- Prototyping, evaluating and debugging SDN applications is hard
  - Increasing scale, diversity, and complexity of apps
  - Will my SDN app behave as expected when deployed in the wild?
  - Does it operate correctly and efficiently at scale?
SDN prototyping and debugging landscape

Simulation
- Custom fluid-flow simulators (e.g., Al-Fares et al., 2009)
- Extensions to packet-level simulators (e.g., Klein and Jarschel, 2013)

Emulation
- Mininet (Handigol et al., 2012)

Live Deployment
- Testbed experiments (e.g., Greenberg et al., 2009)
- Virtual slice of a live network (e.g., with FlowVisor, Sherwood et al., 2010)

Also:
- Debuggers, e.g., ndb (Handigol et al., 2012)
- Static analysis and symbolic execution tools, e.g., VeriFlow (Kurshid et al., 2012), Header space analysis (Kazemian et al., 2012) NICE (Canini et al., 2012)
Goals of our work

• Develop an SDN simulation capability that complements existing development and debugging tools

• A controller API environment to facilitate transition to live environments

• Ability to generate realistic application traffic flows

• Capability to scale up to large networks

• Facilities for detailed debugging and tracing
fs-sdn overview and background

- Designed as extensions to the fs network flow record generator (Sommers et al., INFOCOM 2011)
  - Uses discrete event simulation to drive flow record generation
  - Flowlets instead of packets
  - Accurate to 1 second time scales, way faster than ns2
    - Example from earlier work: speedup of 50x in scenario with 800 new flows/sec
- Written in Python
- SDN extensions leverage and integrate the POX controller platform
  - Provides OpenFlow 1.0 environment
fs-SDN design and implementation

- Integrate POX controller and library code via monkeypatching
  - Key aspects: calls that get or set external state (time, network) and packet/flowlet translation
- Upshot: POX controller modules can be used without modification in fs
  - discovery, spanning tree, l2 learning, hub, l2 pairs, etc., all work out of the box
System evaluation

- Evaluate accuracy and scalability of fs-sdn
- Set up congruent experiments in fs-sdn and Mininet
  - Background traffic: CBR stream or Harpoon flows at two different loads each
  - Linear topologies in 4 configurations of increasing size (up to 100 switches)
  - Simple layer-3 shortest paths controller module
Results: accuracy

- Plots show byte counts per second collected in fs and an equivalent setup in Mininet.
- As topology and/or traffic load increase, measurements collected in Mininet degrade.
  - I/O bottlenecks limit system performance.
Results: speedup

- Tables show fs-sdn execution times for scenarios with 900 simulated seconds
- Mininet takes 900 seconds for each experiment
- pypy interpreter with JIT compiler was used for experiments

<table>
<thead>
<tr>
<th>Load</th>
<th>Tiny</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>6</td>
<td>8</td>
<td>33</td>
<td>72</td>
</tr>
<tr>
<td>High</td>
<td>4</td>
<td>8</td>
<td>31</td>
<td>76</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load</th>
<th>Tiny</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>16</td>
<td>33</td>
<td>104</td>
<td>193</td>
</tr>
<tr>
<td>High</td>
<td>30</td>
<td>62</td>
<td>194</td>
<td>337</td>
</tr>
</tbody>
</table>

UDP CBR traffic

Harpoon traffic (Pareto distr. flow sizes)
Summary and future work

- *fs-sdn* provides a fast and accurate simulation environment for prototyping and debugging SDN/OpenFlow applications
  - (Nearly) seamless transition of controller modules to “real” deployments
  - Code available: [https://github.com/jsommers/fs](https://github.com/jsommers/fs)

Future work

- Complete packet/flowlet translations to truly make the environment seamless
- Better tracing and debugging capabilities
- Improve scalability through parallelizing *fs*
- Is it possible to bridge other (including non-Python) controller platforms?